其他背包dp

2019年4月15日 12:58

-
- 完全背包

一. 基本思想

1. 数据：物品数n、体积v、价值w、背包容量m
2. 变量
   i. 阶段i：处理的第i个物品
   ii. 状态j：当前体积
   iii. 存储的数据：价值w
3. 初值：w[0][0]为0，其余为负无穷（可根据最终结果是不是负数来判断有没有恰能凑出这个体积的货物，或在转移时判断=右边是负值就不转）
4. 转移方程
   i. w[i][j]=max(w[i-1][j], //没选过第i个物品
      w[i][j-v[i]]+w[i]); //再选一次第i个物品
   ii. 循环i：[1,n]（物品编号从1开始时，而且这样不用处理越界）
   iii. 循环j：[v[i],m]（否则会越界）

二. 正序循环法

1. 思想：因为可以无限使用某物品，会改变当前i阶段的值
2. 实现方法：双重正序循环即可
3. ```
   for__(i,1,n)
       for__(j,v[i],m)
           w[j]=max(w[j], //不再加
               w[j-v[i]]+w[i]); //再加一个
   ```

三. 变形

1. 自然数拆分法：将n拆成若干可重复正整数的和，求拆法数量对N的余数
   i. 思想：f[i]存i的拆法数量
   ii. 阶段i：正在判断正整数i
   iii. 状态j：和为j
   iv. 转移：f[j]=(f[j]+f[j-i])%N;
   v. 初值：其他为0，f[0]=1
   vi. ```
   for__(i,1,n)
       for__(j,i,n)
           f[j]=(f[j]+f[j-i])%N;
   ```
2. 分宿舍法：n男m女和k对异性恋情侣，2人间a元，3人间b元，情侣间c元，只有情侣间不能有空床，求最便宜的分法
   i. 思想：用背包d[i]打表i人的最便宜分法，再枚举k对情侣拆了几对即可
   ii. 阶段：2人间或3人间，因为只有两个就不循环了
   iii. 状态j：需要住的人数为j
   iv. 初值：其他为无穷，d[0]=0

v. 转移：d[j]=min(d[j], d[j-2]+a);同理3人间

vi. 注意：因为2人间和3人间不用住满，所以真正的i人最便宜买法可能被存在i+1或i+2了，最简单的解决方法是再逆序循环一次让每个d[i]尝试从min(d[i], d[i+1])转移

vii.
```
memset(d,0x3f,sizeof(d));
d[0]=0;
for__(i,2,n+m+k+5)
        d[i]=min(d[i], d[i-2]+a);
for__(i,3,n+m+k+5)
        d[i]=min(d[i], d[i-3]+b);
rof__(i,n+m+k+4,0)
        d[i]=min(d[i], d[i+1]);
ll ans= 1ll<<60;
for__(i,0,k)
        ans=min(ans, c*i + d[n+k-i] + d[m+k-i]);
printf("%lld\n",ans);
```

-
- 多重背包

四. 基本思想

1. 数据：物品数n、体积v、价值w、个数c、背包容量m
2. 变量
   i. 阶段i：处理的第i个物品
   ii. 状态j：当前体积
   iii. 存储的数据：价值w

五. 直接拆分法

1. 思想：当成01背包来做
2. 初值：w[0]=0，其他为负
3.
```
for__(i,1,n)
        for__(k,1,c[i])
                rof__(j,m,v[i])
                        w[j]=max(w[j]，//不选择第i个物品
                                w[j-v[i]]+w[i]); //选择第i个物品
```

六. 二进制拆分法

1. 循环:while(w>k){ w-=k; k*=2; 对k做01背包}再对剩余的w做01背包
2. 相当于复杂度C拆成了logC
3. 例：求恰装满T容量背包能有多大价值
   i.
```
memset(d, 0x80, sizeof(d));
d[0]=0;
scanf("%d%d",&n,&t);
for_(i,0,n){
        scanf("%d%d",&v,&w);
        int k=1;
        while(w>k){
                rof__(j,t,v*k)
                        d[j]= max(d[j], d[j-v*k]+ k);
                w-=k;
                k*=2;}
```

```
        rof__(j,t,v*w)
                d[j]= max(d[j], d[j-v*w]+ w);}
    cout<<max(d[t],0);
```

ii. 有时v*w>t，可直接套完全背包公式

七. 变形

1. 砝码称重：1，2，3，5，10，20各a[i]个的砝码能称几种重量
   i. 思想：砝码是物品，d[i]存物品权值能不能到i
   ii. 因为只需要记录能不能到，所以不需要循环所有a[i]遍，确认能就行了
   iii. bool d[10005]; //背包

```
int a;              //数量
int w[6]={1,2,3,5,10,20};

int main(){
    d[0]=1;
    for_(i,0,6){
        scanf("%d",&a);
        if(a>0)
            rof__(j,10000,w[i])
                for__(k,1,a)
                    if(d[j-k*w[i]]){
                        d[j]=1;
                        break;}}
    rof_(j,10000,0)
        if(d[j]){
            cout<<j;
            break;
    return 0;}
```

-
- 分组背包

八. 每组只能选一个物品加入

1. 阶段i：从1到m遍历每组
2. 状态j：从v到0遍历每种容量
3. 决策k：遍历i组的每个物品，各做一次01背包转移
4. 只有**j是逆循环**，且**j在k外**，才能保证每种容量，都只更新到1个i组内的物品

-
- 背包搜索

九. 深搜

1. 模版：
   i.
```
void SearchPack(i,cur_v,cur_w){
    if(i>N){
        if(cur_w>best)
            best=cur_w;
        return;}
    if(cur_v+v[i]<=V)
        SearchPack(i+1,cur_v+v[i],cur_w+w[i]);
    SearchPack(i+1,cur_v,cur_w);}
```

2. 最优性剪枝：01背包，加上权值后缀和也比之前的最优解小时直接return
3. 可行性剪枝：装满背包，加上体积后缀和也比之前的最优解小时直接return

4. 顺序：费用大的排前面方便剪枝